Improving the Predictability of Take-off Times with Machine Learning

A case study for the Maastricht upper area control centre area of responsibility

Ramon Dalmau & Franck Ballerini
Network Research Unit (NET)
EUROCONTROL Experimental Centre (EEC)
Brétigny-Sur-Orge, France

Herbert Naessens, Seddik Belkoura & Sebastian Wangnick
Architecture / System Engineering (ENG/ASE), Change Management (DIR/CHG)
EUROCONTROL Maastricht Upper Area Control Centre (MUAC)
Maastricht, The Netherlands

***Abstract*—The uncertainty of the take-off time is a major contribution to the loss of trajectory predictability. At present, the Estimated Take-Off Time (ETOT) for each individual flight is extracted from the Enhanced Traffic Flow Management System (ETFMS) messages, which are sent each time there is an event triggering a recalculation of the flight data by the Network Manager Operations Centre. However, aircraft do not always take-off at the ETOTs reported by the ETFMS due to several factors, including congestion and bad weather conditions at the departure airport, reactionary delays and air traffic flow management slot improvements. This paper presents two machine learning models that take into account several of these factors to improve the take-off time prediction of individual flights one hour before their estimated off-block time. Predictions performed by the model trained on three years of historical flight and weather data show a reduction on the take-off time prediction error of about 30% as compared to the ETOTs reported by the ETFMS.**

***Index Terms*—trajectory prediction, machine learning**

I. Introduction

According to the most likely scenario of the EUROCONTROL's statistics and forecast service (STATFOR), there will be around 16.2 million of flights in Europe in 2040, which corresponds to 57% more traffic than in 2017. This is 1.9% average annual growth per year over the period 2017-2040 [1]. With traffic demand reaching historic levels and projected to continue growing in the years to come, balancing the demand and the capacity has become critically important.

Demand-capacity unbalances are difficult to predict in the pre-tactical phase, mainly because of lack of accurate 4D trajectory information before operations. The introduction of Trajectory-Based Operations (TBO) [2], the cornerstone of the SESAR (Single European Sky ATM Research) programme, will allow the coordination of 4D trajectory predictions and constraints across all operational stakeholders, both during the pre-tactical and the tactical phases of flight. As such, by better predicting and coordinating aircraft trajectories, TBO would improve throughput, flight efficiency and punctuality. In this context, trajectory predictors providing accurate flight information will play a key role to move the current Air Traffic Management (ATM) system towards the TBO paradigm.

However, the lack of detailed and up-to-date flight information as well as the discrepancy between scheduled and flown trajectories created, for instance, by airport operational uncertainties and Air Traffic Control (ATC) clearances are significant weaknesses of state-of-the-art trajectory predictors. The use of historical data by means of Machine Learning (ML) has potential to improve the accuracy of trajectory predictions.

The Maastricht Upper Area Control Centre (MUAC) has developed an Artificial Neural Network (ANN) that is able to predict lateral routes within the MUAC Area Of Responsibility (AoR), based on a subset of flight plan data and the scheduled status of military areas [3]. In contrast to traditional methods for trajectory prediction, which typically rely on kinematic models of the aircraft behaviour and several assumptions about the aircraft intent, this solution relies on historical data and ML. The solution has been deployed January 2018 into the local MUAC Flow and Capacity Management system.

The 4D trajectory of an aircraft, however, consists of the three spatial dimensions plus time as a fourth dimension. That means that any delay is in fact a distortion of the trajectory as much as a change of the lateral route. The current solution is able to predict the lateral routes within the MUAC AoR, but cannot predict the entry times into the AoR. This variability of entry times stems from factors external to MUAC, such as clearances by upstream ATC, general behaviour of aircraft outside MUAC airspace, or take-off time uncertainty at the departure airport. This paper addresses the latter factor.

At present, the Estimated Take-Off Time (ETOT) of each individual flight is obtained from the Enhanced Tactical Flow Management System (ETFMS) Flight Data (EFD), which is regularly updated from the submission of the Initial Flight Plan (IFP) to the Actual Take-Off Time (ATOT). The ETOT reported in the EFD, however, is not accurate enough to provide acceptable trajectory predictability. There exist several causes of the discrepancy between ETOT and ATOT, including congestion and bad weather at the airport, reactionary delays and Air Traffic Flow Management (ATFM) regulations.

MUAC has implemented a method to adjust take-off time estimates as reported in the EFD by evaluating the status of aircraft on the ground (as derived from Automatic Dependent Surveillance-Broadcast data, ADS-B) and keeping track of average taxi-times. This solution improves estimates in a narrow 0-20 min horizon prior to take-off, but cannot address

longer horizons and does not adapt entry time estimates of airborne aircraft. The current solution is based on simple statistics and it does not unlock the full potential of ML.

This paper presents two ML models designed to predict the take-off time of individual flights one hour before their Estimated Off-Block Time (EOBT): a Gradient Boosted Decision Trees (GBDT) and an ANN. Both models were trained on three years of EFD for flights crossing the MUAC's AoR and METeorological Aerodrome Reports (METARs) of their corresponding departure airports. The quality of the predictions performed by these two models was measured against the take-off time accuracy of the ETOTs reported by the ETFMS.

II. State of the art

The prediction of take-off times has been a subject undergoing intense study in the last decade. Most state-of-the-art studies address this problem by predicting the departure delay.

The problem of predicting departure delays can be classified according to the granularity of the system for which the delay is being predicted (network-wide, airport, origin-destination pair or individual flight), the look-ahead time of the prediction (strategic, pre-tactical or tactical), the type of model used to solve the problem, and the features considered by the model. An excellent review of the state-of-the-art on flight delay prediction can be found in Ref. [4], which describes how this problem is typically addressed and which compares several methods that have been successfully implemented.

Reference [5] compared the performance of different approaches to predict flight delays at a network-wide level. For instance, Ref. [6] proposed a model capable to estimate and analyse flight delays and cancellations in the United States air traffic network by using publicly available data from social media to train regressors build on decision trees. The outputs of the model include the total minutes of departure and arrival delays in the network as well as the number of cancelled flights, among other metrics aggregated at network-wide level.

Reference [7] compared several binary classifiers build on decision trees to predict whether an individual flight would be delayed by more than 15 min or not. The proposed model requires only weather and calendar data to predict delayed flights with an accuracy up to 80%. However, the model was trained and evaluated only for a specific origin-destination pair and does not provide the specific flight delay in minutes.

A similar approach was proposed by Ref. [8], which implemented random forest classifiers and regressors capable to predict departure delays (either status or specific value, respectively) on selected origin-destination pairs with a look-ahead time ranging from 2 to 24 hours. The proposed models consider the arrival and/or departure delay states of the most influential airports and origin-destination pairs in the network as features. Results on unseen data showed a median error around 20 min, considering a look-ahead time of 2 hours. Reference [9] also attempted to solve the delay prediction problem by using a deep ANN architecture. The proposed model works by initially predicting the coarse delay status of the whole air traffic network. Then, the fine-grained delay state of each individual flight is classified by feeding the delay status of the network along with several flight-specific features.

Most of the aforementioned studies, however, attempted to predict delays aggregated at a very coarse level (e.g., the whole air traffic network or particular airports or routes) or to classify an individual flights into a binary class (either delayed or not). Only few works aimed to predict the accurate delay of a flight in minutes. State-of-the-art results on solving this challenging problem can be found in Ref. [10], which recently proposed a two-stage model: the first stage consists of a classifier that predicts the occurrence of flight delays (i.e., detects whether the flight will be delayed or not), and the second stage consists of a regressor that predicts the value of the delay with a Mean Absolute Error (MAE) around 8 min.

Recently, [11] proposed Deep Belief Network (DBN) that considers novel features, such as the delay on the previous leg of the same airframe or the demand level (congestion) at the airport, to predict delays of particular flights with a MAE of 8.5 min. Results showed that these features are of high importance to improve the accuracy of flight delay predictions.

III. Sources of data

The models proposed in this paper were trained on historical flight traffic and weather data. The following sections describe the sources of these two different types of data, respectively.

A. Historical flight traffic data

Historical traffic have been obtained from the ETFMS, which collects and distributes relevant flight information to the Air Navigation Service Providers (ANSPs) for flights entering their airspace and to Aircraft Operator (AOs) for flights with flight plans submitted to the NMOC. As such, the ETFMS captures all traffic crossing the Network Manager's (NM) area of operations, regardless of the origin and destination.

The ETFMS provides predicted information about the flight prior to take-off including the EOBT; the ETOT; the planned taxi-in time; the planned route to the destination airport including the Standard Instrumental Departure (SID); the airframe type and registration number; as well as the ATFM delay and the list of regulations to which the flight is subject (if any). The ETFMS also provides actual and predicted information after take-off including the Actual Off-Block Time (AOBT); the ATOT; airborne position updates; as well as the Estimated Time of Arrival (ETA). The Actual Time of Arrival (ATA) is also provided once the flight terminates.

B. Historical weather data

Historical weather data have been obtained from METeorological Terminal Aviation Routine Weather Reports (METARs). METARs are typically generated once an hour at airports or at weather observation stations. A standard METAR includes information about the temperature, pressure, dew point, wind direction and speed, precipitation, cloud cover and height, as well as visibility and ceiling. It may also include information about the presence of specific weather phenomena such as precipitation and obscuration type and intensity.

IV. Content of the Dataset

Given a dataset $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$ composed of N training examples of the form $\{(x_1, y_1), ..., (x_N,\ y_N)\}$, where $\mathcal{X}$ is the *features* space and $\mathcal{Y}$ is the *target* space, a ML algorithm aims to find the best function $g : \mathcal{X} \to \mathcal{Y}$ to predict $y \in \mathcal{Y}$ for any $x \in \mathcal{X}$. In order to measure how well a function g fits $\mathcal{D}$, a *loss* function $L : \mathcal{Y} \times \mathcal{Y} \to \Re$ is defined. For the i-th example $(x_i,\ y_i)$, the loss of predicting $\hat{y}_i = g(x_i)$ is $L(y_i, \hat{y}_i)$.

The function that better fits $\mathcal{D}$ can be found by minimising the *risk*, which can be considered as the expected loss of g:

$$R(g) = \frac{1}{N} \sum_{\mathcal{D}} L(y_i, \hat{y}_i) = \frac{1}{N} \sum_{\mathcal{D}} L(y_i, g(x_i)). \qquad (1)$$

Next sections present the target predicted by the ML models and the features selected to accomplish this task, respectively.

A. Output target

Let us define t_i^{pred} as the time at which the take-off time of flight i has to be predicted by the model g. In this paper, t_i^{pred} corresponds to the 1 hour before EOBT, or the Controlled Off-Block Time (COBT) if the flight is subject to regulation.

The problem of predicting the take-off time of a flight is equivalent to anticipate the take-off time prediction error of the ETFMS. Accordingly, the target variable for each example, here corresponding to a certain flight, $(x_i,\ y_i)$ is the difference between ATOT and ETOT reported by the ETFMS at t_i^{pred}:

$$y_i(t_i^{pred}) = \text{ATOT}_i - \text{ETOT}_i(t_i^{pred}), \qquad (2)$$

Note that for each flight i there is one single ATOT, while the ETOT is time-dependent. The proposed system works by predicting $\hat{y}_i$ and then correcting the ETOT of the ETFMS:

$$\widehat{\text{ETOT}}_i(t_i^{pred}) = \text{ETOT}_i(t_i^{pred}) + \hat{y}_i(t_i^{pred}). \qquad (3)$$

B. Input features

The whole set of features listed in the following sections have been divided by theme (flight, congestion, delay state, weather, and calendar). In turn, each feature has been further classified as categorical (discrete) or continuous, and whether it is static or dynamic during the progress of the flight.

ETFMS messages are sent every time there is an event triggering a recalculation of the flight data by the Network Manager Operations Centre (NMOC). Accordingly, several messages can be sent for each flight i from the submission of the initial flight plan at t_i^{IFP} to prediction of the take-off time at t_i^{pred}. This implies that, in reality, x_i is not a vector of features but a sequence of vectors $\boldsymbol{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,N_i})$, where the N_i is the number of messages sent by flight i from t_i^{IFP} to t_i^{pred}, and the length of each element $x_{i,j}$ in that sequence is equal to the number of features. Therefore, the following features are computed for each message of every single flight. For most of these features, the name explains itself. For those that not, a description is provided.

1) Flight features: These features include basic flight information and also the most up-to-date status of the flight and its previous and next legs at t. Some of these features, such as the ATFM delay, aircraft operator or departure and destination airports, can be directly extracted from the EFD fields; some others, such as the duration of the flight or the time since the current airframe was allocated to the flight, require some data manipulation. Table I lists the features included in this set.

TABLE I. Features related to the flight

Feature j	Type	
Available turn-around time Available turn-around time for the next leg Duration of the flight Duration of the previous leg ATFM delay Time to EOBT EOBT delay w.r.t IFP Target Off-Block Time (TOBT) - EOBT Target Start-up Approval Time (TSAT) - EOBT Time from IFP Time from airframe allocation Taxi-in time Time to take-off for the previous leg Time from previous leg message Delay of the previous leg w.r.t IFP	Numerical	Dynamic
Aircraft type Aircraft operator of the previous leg Aircraft operator of the next leg Departure airport of the previous leg Destination airport of the next leg ATM flight status ATM flight status of the previous leg Event that triggered the message Event that triggered the previous leg message Last letter of the SID Traffic volume of most penalising regulation	Categorical	Dynamic
Departure airport Destination airport Country of departure Region of departure Alliance of the airline Aircraft operator		Static

The turn-around time is defined as the time required to unload an aircraft after its arrival at the gate and to prepare it for departure again. The turn-around time available to the flight as expected at time t is defined as the difference between the most updated information about the time of arrival (either actual if landed, controlled if regulated or estimated otherwise) of the previous leg and the current EOBT of the flight. Note that this feature is dynamic because both EOBT and time of arrival of the previous leg might change with t.

The time since the IFP was sent is also considered in order to identify flights which have not submitted the flight plan 3 hours before EOBT, also known as *late filers*. The Computer Assisted Slot Allocation (CASA) algorithm, which is in charge of assigning ATFM slots to regulated flights, penalises late filers by giving them less chances of improving their slots.

Other features included in this set are the ATM flight status and the type of event that triggered the most recent message for the current, previous and/or next legs of the flight.

These features are low-cardinality categorical variables, which possible values and description can be found in Ref. [12].

The TOBT is defined as the time that an AO expects that the aircraft will be ready to push back immediately upon reception of clearance from the ATC. The TSAT is the time provided by ATC taking into account TOBT, CTOT and/or the traffic situation that an aircraft can expect to receive push back approval. The differences between these times (if available) and the EOBT have been also included as explanatory variables.

Finally, knowing the runway configuration of the airport that will be active at ETOT is expected to improve the predictive power of the algorithm. Unfortunately, this information is not transmitted by the ETFMS. In most of Europe, however, the SID is named according to the final waypoint of the procedure, followed (optionally) by a version number that is increased each time the SID is updated, and a single letter that designates the runway. Accordingly, a proxy of the runway that the aircraft plans to use for take-off is the last letter of the SID.

2) Expected congestion level: These features seek to capture the congestion level at the departure airport at the expected take-off time. In order to accomplish that, they include the planned number of arrivals and departures near ETOT. Note, however, that only information available at t can be used to compute these features. Table II lists the features included in this set. Features related to congestion at the destination airport at the ETA are not included because feature importance analysis indicated that their impact on the prediction is marginal.

TABLE II. FEATURES RELATED TO CONGESTION AT THE AIRPORT

Feature j	Type	
# of departures at ETOT $\pm$ 15 min # of departures at ETOT $\pm$ 30 min # of departures at ETOT $\pm$ 60 min # of arrivals at ETOT $\pm$ 30 min # of arrivals at ETOT $\pm$ 15 min # of arrivals at ETOT $\pm$ 30 min	Numerical	Dynamic

3) Delay state features: These features aim to represent the current delay state of the departure airport at the current t. In this paper, the number of aircraft with departure/arrival delay in certain intervals are used as delay state features. Table III lists the features included in this set.

TABLE III. FEATURES RELATED TO DELAY STATE OF THE AIRPORT

Feature j	Type	
# of departures/arrivals with delay in (-30, -15] min # of departures/arrivals with (-15, -5] min of delay # of departures/arrivals with (-5, 0] min of delay # of departures/arrivals with (0, 5] min of delay # of departures/arrivals with (5, 15] min of delay # of departures/arrivals with (15, 30] min of delay # of departures/arrivals with (30, 60] min of delay	Numerical	Dynamic

Note that the features shown in Table III are computed for three different time windows, considering the flights that departed or arrived at the airport 1, 3 and 6 hours before t.

4) Weather features: This set includes the most relevant weather information at the departure airport, which is extracted from the most recent METAR at time t. These features are listed in Table IV. Weather information at the destination airport was also discarded after a feature importance analysis.

TABLE IV. FEATURES RELATED TO WEATHER AT THE AIRPORT

Feature j	Type	
Temperature Visibility Ceiling Wind direction Wind speed Wind gust	Numerical	Dynamic
Precipitation Cloud cover Cloud type	Categorical	

5) Calendar features: Finally, basic calendar features such as the hour of the day, the day of the weel or the month of the year are also included in the model as categorical inputs, aiming at capturing temporal trends and seasonalities.

V. MACHINE LEARNING MODELS

Two different models have been evaluated in this study: Gradient Boosting Decision Trees (GBDT) and Recurrent Neural Networks (RNN). The following sections describe the architecture of these two models, respectively. In both cases the MAE was minimised to fit the model g to $\mathcal{D}$, that is:

$$R(g) = \frac{1}{N}\sum_{\mathcal{D}} L(y_i, g(x_i)) = \frac{1}{N}\sum_{\mathcal{D}} |y_i - g(x_i)| \quad (4)$$

The following sections assume that the reader knows basic concepts of ML such as ensembles, decision trees, dense and embedding layers as well as dropout mechanism. If this were not the case, a highly recommended reference is [13].

A. Gradient Boosting Decision Trees (GBDT)

Ensemble methods are a set of machine learning techniques that construct a strong learner from a number of weak learners. *Boosting* is a well-known ensemble method which consists of iteratively training a sequence of weak learners, where the training examples for the next learner are weighted according to the accuracy of the previously constructed learners. GBDTs build a decision tree learner at a time by fitting the gradients of the residuals of the previously constructed decision trees.

GBDTs are very popular models in the field of ML due to their state-of-the-art performance in many tasks. Even if conventional implementations of GBDT such as XGBoost [14] are appropriate for a lot of practical applications, their efficiency and scalability are still deficient when number of features and/or the number of examples in the train set is large.

LightGBM [15] is another GBDT implementation that implements Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) to alleviate this issue.

LighGBM, however, is not explicitly designed to deal with sequential data. Remember that, in this paper, each example x_i is a sequence of vectors, where each vector corresponds to a message sent by flight i and includes the features listed

in Section IV; and the target y_i is the prediction error of the ETFMS at t_i^{pred} for the associated flight. In this experiment, only the last message of each flight i has been used to train the LightGBM model, ignoring all messages previous to t_i^{pred}. This means that the LighGBM model only sees the most recent information of the flight before performing the take-off time prediction, thus is not able to capture information that may be included in the evolution of the flight from t_i^{IFP} to t_i^{pred}.

B. Artificial Neural Network

The aim of this model is to consider the whole sequence of messages sent for each flight before predicting its take-off time, hoping that previous messages will include beneficial information to capture the ETFMS errors. In order to accomplish that, Recurrent Neural Networks (RNN) have been used, which are specially designed to deal with sequential data.

Roughly speaking, a RNN has a looping mechanism that allows information to *flow* from one time sample to the next one. This information is the hidden state, which is a representation of previous inputs. At each time sample, the RNN takes the inputs vector and updates the hidden state.

One must be cautious when designing a RNN that takes static and dynamic features. The most reasonable approach consists of conditioning the RNN on the static features. In order to accomplish that, the hidden state of the RNN at the first time step is modelled as a function of the static features. However, the hidden state of the RNN need to have the shape of the dynamic features. The approach taken by [16], [17] has been implemented in this paper to satisfy this requirement. For each training example, the vector of static features is transformed with a dense layer to get it into the same shape as the hidden state of the RNN. Then, for the very first t, the transformed vector is set as the hidden state of the RNN.

Another issue to address is how to deal with categorical variables. One-Hot encoding is a popular method for converting categories into continuous variables. Unfortunately, one-hot encoding of high-cardinality features often results in an unnecessary amount of computational resources. Furthermore, this technique treats different values of categorical variables completely independent of each other, often ignoring the potentially informative interactions between them.

An alternative to one-hot encoding is to embed each categorical variable. An embedding automatically learns the representation of a categorical variable in a multi-dimensional space. The result of the embedding is a vector of continuous values in which categories with similar effect to the target are close to each other. In this paper, each categorical feature is passed through an independent embedding. Dynamic embeddings are concatenated with the numerical features (all of them being dynamic). The RNN is conditioned with the transformed vector of static embeddings, and is fed with the dynamic features. Several RNNs can be stacked, and the output of the last RNN is passed through a stack of dense layers potentially with dropout in-between them to deal with overfitting. Since this model is performing a regression task, the last dense layer must have one single neuron and linear activation function.

VI. Results

The two models presented in Section V, which consider the features listed in Section IV as inputs to anticipate the take-off time prediction error of the ETFMS, were trained on flight traffic and weather data for three years (from January of 2016 to December of 2018). From the whole list of flights captured by the ETFMS, only those crossing the MUAC airspace were considered for this experiment. From this subset of flights additional filtering was performed to remove suspended and cancelled flights, and also those flights with an absolute take-off time prediction error above the 99th percentile (which were considered outliers). The total number of instances after filtering was 4.8M, from which 70% were used for training the model, 10% for early-stopping and hyperparameters tuning, and the remaining 20% for testing its accuracy on unseen data.

A. Optimal hyperparameters of the models

The selection of correct values for the model's hyperparameters could boost its performance. In this paper, Tree-structured Parzen Estimator (TPE) has been used to optimise the hyperparameters. Roughly speaking, TPE is Bayesian optimisation method which algorithm selects the hyperparameters and corresponding values to evaluate in the next iteration based on the distribution of the previous results.

1) LightGBM: There are many hyperparameters in this model, controlling both the entire ensemble and individual decision trees. Table V shows the optimal values of the most influencing ones, resulting from the TPE optimisation.

TABLE V. Hyperparameters for the LightGBM model

hyperparameter	value
num_leaves	512
num_trees	512
max_depth	32
learning_rate	0.05

2) Artificial Neural Network: The hyperparameters of the ANN include those specific to the model's architecture, such as the minimum size of the embeddings, the type of RNN cell used in the recurrent layers, the number of RNN and Dense layers as well their corresponding number of neurons and activation function; and those specific to the training process, such as the learning rate, the batch size or the number of epochs. These hyperparameters are shown in Table VI.

TABLE VI. Hyperparameters for the ANN model. Notation for layers: # of neurons-activation function-dropout rate

hyperparameter	value
RNN cell type	LSTM (Long Short Term Memory)
Stacked RNNs layers	256-tanh-0 → 256-tanh-0
Stacked Dense layers	64-ReLu-0.6
optimiser	Adam
batch_size / learning_rate	16 / 1e-4

B. Performance of the models

Table VII shows the performance of the current ETFMS evaluated in the test set, as well as the same metrics when correcting its ETOTs with the two models proposed herein.

TABLE VII. PERFORMANCE METRICS IN THE TEST SET

Model	Performance metric				
	MAE	σ	Q1	Q2	Q3
ETFMS	10 m 10 s	10 m 57 s	3 m 0 s	6 m 37 s	13 m 0 s
LightGBM	7 m 8 s	8 m 0 s	2 m 11 s	4 m 50 s	9 m 5 s
ANN	7 m 22 s	8 m 19 s	2 m 18 s	5 m 3 s	9 m 15 s

According to Table VII, the current ETFMS predicts the take-off time of flights crossing the MUAC AoR with a MAE sightly above 10 minutes. Remember that in this paper t_i^{pred} corresponds to 1 hour before EOBT. Both ML models are able to correct the ETFMS predictions and reduce its MAE to roughly 7 minutes, which represents an improvement of 30%.

The standard deviation, which measures dispersion around the mean value, is also reduced around 30% if compared to the current ETFMS. A smaller standard deviation means greater consistency, predictability and quality of the predictions.

These results suggest that the selected features are able to capture several systematic errors of the ETFMS and considerably improve the quality of take-off time predictions, yet there are still a lot of missing factors leading to mismatches between ETOT and ATOT that have not been included in the model (e.g, passenger's connections). After all, however, there exists random effects and intrinsic noise of the system that even the most advanced ML models will not be able to capture.

Results also show that the performance of the LightGBM model is just as good as that of the ANN, suggesting that the information of the whole sequence of messages for each flight does not add clear benefit to the prediction, that the ANN cannot extract it, or that this sequential information has already been included satisfactorily by the selected features.

C. Features importance

An important question in the field of ML is why the model made a certain prediction given the input features. In many applications, answering this question could be as important as the accuracy of the prediction itself. In point of fact, being able to understand the rules beneath the decisions of the model does not only increases the trust on its predictions, but also allows to figure out how the process being modelled works, as well as provides intuition on how to improve the results.

Additive Feature Attribution Methods (AFAM) are a set of local methods that assign an importance value ϕ_j to each feature j. The sum over ϕ_j approximates the output $g(x_i)$ of the original model. A well know AFAM method is the Shapley method. The Shapley value ϕ_j for a given feature j is computed by calculating the prediction of the model without the feature j, calculate the prediction of the model including that feature, and then calculating the difference. The effect of removing a feature, however, depends on the other features in the model. Accordingly, the preceding differences are computed for all possible subsets $\mathcal{S} \subseteq \mathcal{X} \setminus \{j\}$.

Figure 1 shows the top 10 most important features of the LightGBM model. This type of graph aggregates Shapley values for all the features and all examples in the test set. The y-axis indicates the feature name, in order of importance from top to bottom. Each dot in the x-axis shows the impact of the associated feature on the final prediction for one example, with positive (resp. negative) values meaning that the impact of the feature was to predicted a late (resp. early) take-off time if compared to that reported by the ETFMS. The gradient color indicates the value for that feature in the corresponding example. Note that Shapley values have the unit of minutes.

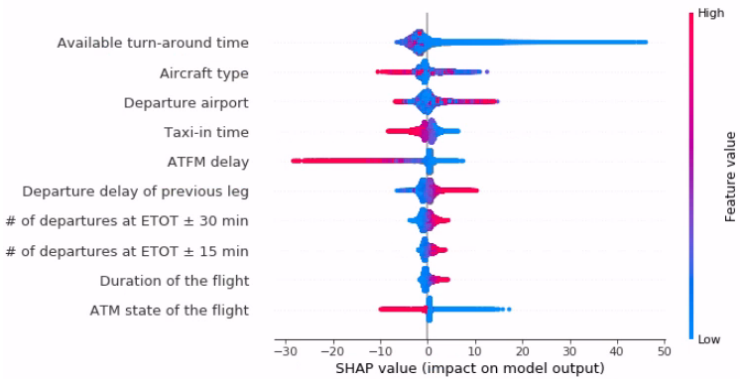


Figure 1. Top 10 most important features

Results show that the time available to perform the turn-around is the main contributor to discrepancies between the ETOT predicted by the ETFMS at t_i^{pred} and the ATOT. As expected, low values of that feature are associated with highly positive take-off time prediction errors. These results suggest that the model is able to discriminate whether the time available to perform the turn-around will cause a delay or not.

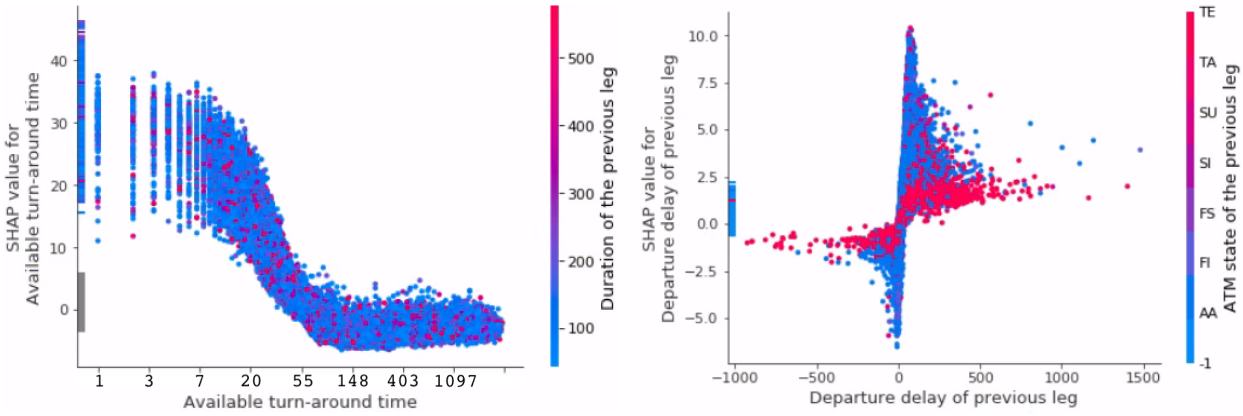
Another conclusion that can be drawn from Fig. 1 is that high taxi times are typically over-estimated. In other words, when the planned taxi time is high, ETFMS tends to report a later take-off time than the one that will be executed in reality. The inverse behaviour is observed for low taxi time values.

Figure 1 also shows that the higher the delay of the previous leg, the later the aircraft is expected to take-off if compared to the value reported by the ETFMS. This means that the model proposed in this paper is able to capture reactionary delays.

Next, the impact that some of the features have on the predictions will be thoroughly analysed. In the Figures 2-5, vertical dispersion represents interaction effects, and the grey ticks along the y-axis are missing values.

1) Flight features: Figures 2a and 2b show the dependence of the target with respect to the time available to perform the turn-around and the departure delay of the previous leg.

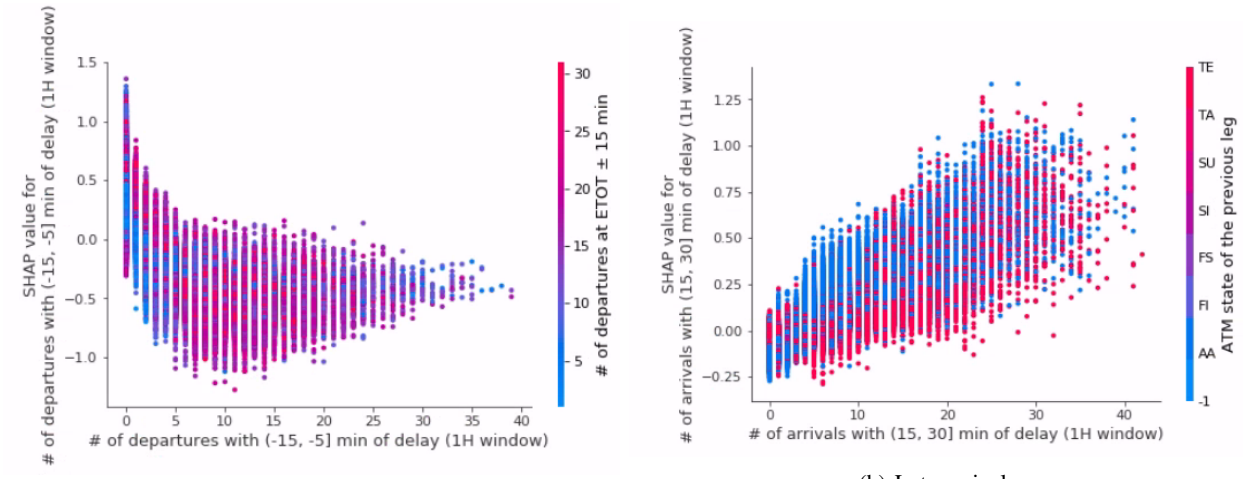
According to Fig. 2a, when the time available to perform the turn-around is below a given threshold, which might depend on the airport and the airline, among other factors, the impact of this feature on the prediction rapidly increases and reaches values up to 40 minutes. For values above the threshold, the



(a) Available turn-around time

(b) Delay of previous leg

Figure 2. Dependence of flight features



(a) Early departures

(b) Late arrivals

Figure 3. Dependence of delay state features

importance of this feature drops significantly. Interestingly, when the value of this feature is high, its importance clearly interacts with the duration of the flight.

Figure 2b shows that the delay of the previous leg of the flight is highly correlated with its Shapley value. As expected, for a given delay, the importance of this feature is higher when the previous leg is still airborne or has not take-off yet.

2) Delay state features: Figures 3a and 3b show the dependence of the target variable with respect to the number of early departures and the number of late arrivals in the last hour, respectively. In this analysis, the intervals (-15, -5] and (15, 30] have been selected for illustrative purposes.

According to Fig, 3a, a low number of early departures suggests that the concerned flight will probably depart later than expected, while high values of this feature indicate that the ETOT predicted by the ETFMS is overestimated. Conversely, Fig. 3b shows that the impact of the number of late arrivals increases as does its value. These results indicate that the model is able to identify the delay status of the airport.

3) Expected congestion level: Figure 4 shows the dependence of the target with the # of departures at EOBT$\pm$15.

According to Fig. 4, when the # of departures near ETOT is low, this feature suggests that the flight will depart earlier than the ETOT of the ETFMS. There exists a threshold above which the effect of this feature is to predict some delay. In general, this threshold is near 12, but depends on other factors such as the airport and the weather conditions. Interaction results also show that this feature is more important for short flights. In other words, when there is congestion, airports seem to give priority to long-haul flights. On the other hand, in case of low demand, long-haul flights do not depart earlier.

4) Weather features: Figures 5a and 5b show the dependence of the target with the temperature and the visibility.

According to Fig. 5a, negative temperatures are typically related with positive take-off time prediction errors. Conversely, positive temperatures do not significantly influence the prediction. As expected, Fig. 5b shows that the lower the visibility, the later the flight is expected to depart.

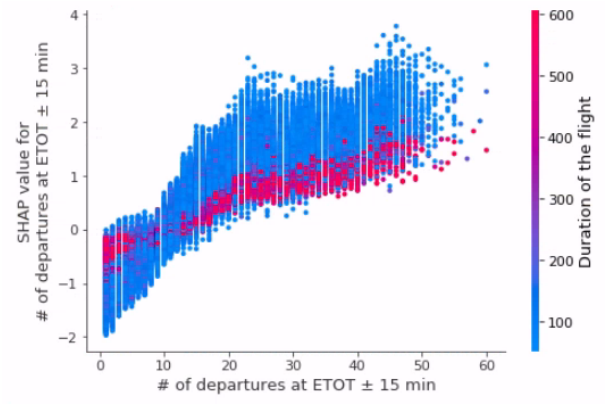
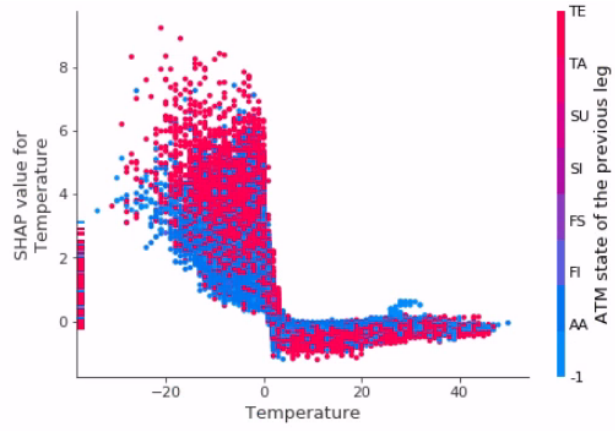
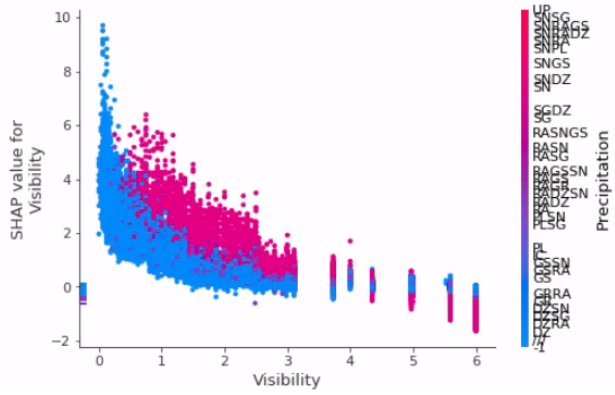


Figure 4. # of departures at ETOT ± 15 min



(a) Temperature



(b) Visibility

Figure 5. Dependence of weather features

VII. Conclusions

Results from predictions performed by a Gradient Boosted Decision Trees (GBDT) and an Aritificial Neural Network (ANN) trained on three years of historical flight and weather data showed a reduction of the take-off time prediction error around 30%, if compared to the predictions of the current Enhanced Traffic Flow Management System (ETFMS). Yet, other missing features would definitely boost their predictive power. The missing features correspond to those related to passengers boarding status, aircraft maintenance and mechanical problems, crew schedules, passengers connections, runway configuration of the airport, the occupancy of parking stands as well as other information specific to airlines and airports. At present, this kind of data is either not shared by airlines and airports due to confidentiality policies or prohibitively difficult to obtain. Results from this study shall encourage all operational stakeholders to progressively share the data that could be beneficial to improve the quality of the existing ML models or to develop new ones in favour of the overall Air Traffic Management (ATM) system's performance.

References

[1] EUROCONTROL, "Challenges of growth 2013: Summary report," June 2018.

[2] International Civil Aviation Organization, *TBO Concept*, ICAO, Montreal, QC, Canada, 2015.

[3] H. Naessens, T. Philip, M. Piatek, K. Schippers, and R. Parys, "Predicting flight routes with a Deep Neural Network in the operational Air Traffic Flow and Capacity Management system," EUROCONTROL Maastricht Upper Area Control Centre, Maastricht Airport, The Netherlands, Tech. Rep., December 2017.

[4] A. Sternberg, J. de Abreu Soares, D. F. de Carvalho, and E. S. Ogasawara, "A review on flight delay prediction," *ArXiv*, vol. abs/1703.06118, 2017.

[5] K. Gopalakrishnan and H. Balakrishnan, "A Comparative Analysis of Models for Predicting Delays in Air Traffic Networks," in *12th USA/Europe Air Traffic Management Research and Development Seminar (ATM2019)*. Seattle, WA: FAA/EUROCONTROL, 2017.

[6] P. Monmousseau, D. Delahaye, A. Marzuoli, and E. Feron, "Predicting and Analyzing US Air Traffic Delays using Passenger-centric Data-sources," in *13th USA/Europe Air Traffic Management Research and Development Seminar (ATM2019)*. Vienna, Austria: FAA/EUROCONTROL, 2019.

[7] S. Choi, Y. J. Kim, S. Briceno, and D. Mavris, "Prediction of weather-induced airline delays based on machine learning algorithms," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. Sacramento, CA: IEEE, Sep. 2016, pp. 1–6.

[8] J. J. Rebollo and H. Balakrishnan, "Characterization and prediction of air traffic delays," *Transportation Research Part C: Emerging Technologies*, vol. 44, pp. 231–241, 2014.

[9] Y. J. Kim, S. Choi, S. Briceno, and D. Mavris, "A deep learning approach to flight delay prediction," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. Sacramento, CA: IEEE, Sep. 2016.

[10] B. Thiagarajan, L. Srinivasan, A. V. Sharma, D. Sreekanthan, and V. Vijayaraghavan, "A machine learning approach for prediction of on-time performance of flights," in *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*. St. Petersburg, FL: IEEE, Sep. 2017.

[11] B. Yu, Z. Guo, S. Asian, H. Wang, and G. Chen, "Flight delay prediction for commercial air transport: A deep learning approach," *Transportation Research Part E: Logistics and Transportation Review*, vol. 125, pp. 203–221, 2019.

[12] Hans Koolen and Ioana Coliban, *Flight Progress Messages Document. Edition No. : 2.501*, Eurocontrol, Brussels, Belgium, 2019.

[13] A. Geron, *Hands-on machine learning with Scikit-Learn and TensorFlow*.

[14] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794.

[15] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 3146–3154.

[16] A. Karpathy and F. Li, "Deep visual-semantic alignments for generating image descriptions," *CoRR*, vol. abs/1412.2306, 2014.

[17] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," *CoRR*, vol. abs/1411.4555, 2014.